contact 3

the user group newsletter for sept. 1978 apple computer inc.

INTRODUCING APPLE SOFTWARE BANK CONTRIBUTED PROGRAMS

With this issue we enclose the first catalog of user contributed software, and express our thanks to all those both inside and outside of Apple Computer whose efforts made it possible.

The purpose of the contributed section of the software bank is to allow APPLE users to share programs among themselves easily and at very low cost. By setting up a system to make this possible, we at APPLE hope to bring about a cross-fertilization of the user community, with better programs and applications ideas as the result.

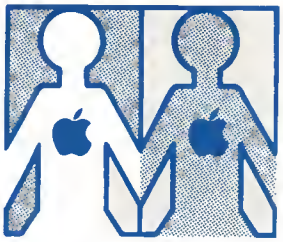
Here is a brief description of the program. More details will be found in the enclosed catalog.

As APPLE receives contributed programs, they are screened. The necessary documentation is assembled, and written catalog descriptions are produced for each program accepted. The author is then sent a $25 APPLE merchandise certificate.

Periodically, the available programs will be assembled onto diskettes, bulk duplicated, and supplied to APPLE dealers at a modest cost. The dealers will have unlimited reproduction rights to the programs on each diskette. As each diskette is released, APPLE will distribute a catalog of its contents to you along with CONTACT. When you see something you like, just call or visit your local dealer for a copy.

The library is a resource. In it you will find some programs you can use as is, some you can improve upon, and some that may not interest you at all. That's as it should be. Use what you can, send in your improvements, and let's work together to build a super library!

Oh, about the Bard on the cover. His image is one of many graphics demonstrations available on the first set of Software Bank diskettes. It was produced with a modified facsimile machine that digitized the original artwork, and a graphic processing program on a PDP-11/03 that reduced the image to an 8K byte database. It can now be displayed by the Apple in the Hi-Res Graphics mode.



LOCAL USER GROUPS

...and the beat goes on

With this issue of CONTACT, we're pleased to add five more groups to the rapidly growing list of APPLE user groups:

CALIFORNIA –

APPLE CORE
Scott Kamins
Box 4816
San Francisco, CA 94101

OREGON –

STEMS FROM APPLE
Ken Hoggatt
9195 S.W. El Rose Court
Tigard, OR 97223
(503) 639-5505 (home)
(503) 644-0161, x6136 (work)

TEXAS –

APPLE SEED
The Computer Shop
6812 San Pedro
San Antonio, TX 78216
Bill Hyde
(512) 828-0553

(No name given for this group)
c/o Computer Solutions
Suite 124A, 9200 Broadway
San Antonio, TX 78217
Philip W. Jackson
(512) 828-1455
(800) 292-7652 (toll free)

ARKANSAS –

Interested in joining or forming an APPLE user group? Contact C. Johnson, c/o dataCope, 5706A W. 12th St., Little Rock, AR 72204. (501) 666-8588.

If you're interested in forming or joining an APPLE user group in your area, call or visit your local APPLE dealer; he'll be glad to help. And if you know of an APPLE group that's not yet been listed in this newsletter (see CONTACT No. 1 and No. 2 for the listings of the groups that we know about), write us here at APPLE. Direct your note to Phil Roybal, marketing manager.

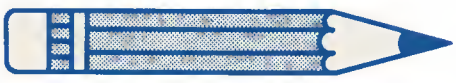
LOOKING AHEAD

...Catalogs, Catalogs

With this issue of CONTACT we've included an APPLE catalog and a price list. If they've been swiped by the time you read this, or if you'd like another set "for a friend," just drop us a line – we'll be glad to send you the literature that you request.

Note, too, that this mailing includes the first catalog of User Contributed Software, with instructions on how to get it!

EDITORIAL



...on user groups.

by Jim Hoyt (Silicon Apple Programming Society)

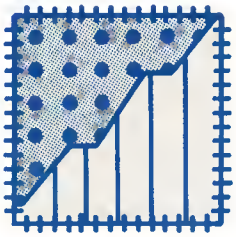
The name says it quite explicitly. USER GROUP: individuals drawn together by a common interest in getting maximum benefit from their computer investment.

Member benefits are manyfold. For the newcomer, programming techniques are explained, the inner workings of the computer made more clear. The answers to many beginner's questions are found right here.

More experienced members can get help or help others with software and hardware design. (Some of the things being done would amaze you: APPLE controlled cassette I/O and lower-case displays to name only two!)

All members benefit from the group's library. There is probably no better place to get low cost software or that back issue of the magazine you need. And many members enjoy newsletters published by their group which supplement our own with hints, local news, and application ideas.

The growth of the personal computer industry depends on you, the USER. Join a group. If there is no local group, consider starting one. A group ties it all together!



PATCHES AND PROGRESS

APPLESOFT II and DOS ...squashing a bug

If (in the DOS) you try to load a program that had been saved under RAM APPLESOFT II, but you are now using the ROM card version, your program will not run correctly. To get around this problem, load your program and then type

CALL 54514

and your program will be correct.

Similarly, a program that was saved onto diskette from the ROM card version and later loaded under RAM APPLESOFT II will also cause problems. Simply type

CALL 3314

and your program will be correct. You can now save it onto tape.

The earliest production DOS had a problem: when in APPLESOFT II, any Read or Write statements with line numbers of 256 or higher would be ignored. To solve this problem on any disks you may INIT in the future, bring up the DOS, remove the Write Protect sticker from your system's master diskette, and then type:

```
>PR#n (boot your system)
>BLOAD RAWDOS
>(hit Reset)
*25D6:4C D5 3F
*25DC:2E
*3FD5:E8 F0 1 60 4C DD 25
*3D0G
>BSAVE RAWDOS, A$1B00,
  L$2500
```

Any new masters created from this original master diskette will now work properly.

Two corrections to CONTACT 2, June '78

There is one correction to be made in each of two program listings in the HOW TO section of the last issue of CONTACT.

– The CONVERT program on Page 6: statement 251 should read
251 IF T<>0 THEN NEXT B:B=B+1: IF B<E THEN 40

– The HOW TO SET LOMEM WITHOUT HARDLY TRYING program on page 7: we should have tried harder. We now hasten to point out that, in statement 40, *B* cannot be a variable, since setting the new LOMEM will destroy the variables (as we ourselves said, in the paragraph just preceding the listing). The concept works, but make sure that you POKE in *numbers* in that line, *not* variables.

Cockpit errors and DOS

A high percentage – 75 per cent, to be precise – of disk errors found so far are due to users trying to run APPLESOFT II by typing "RUN APPLESOFT." This seems reasonable, but APPLE II doesn't see it quite that way. As a result (among other things) you cannot reload programs saved on disk.

To get things to come out right – all pointers where they should be, etc. –, from Integer BASIC simply type

FP (Carriage Return)

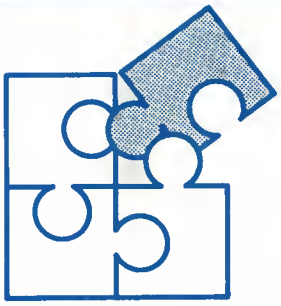
and you'll bring in APPLESOFT II, pointers and all.

To get back to Integer BASIC simply type INT.

About that Communications Card manual . . .

We've put together some additional information that will be needed by many of you using our Communications Card. Included are a routine for use with low-speed printers, information on setting the card's status register, information on setting up a 1200-baud data rate, and a number of other interesting tidbits, as well.

We will be including this addendum to the card manual with all future card shipments, and we'll also be distributing it to all our dealers. So if you already have an APPLE Communications Card, just call your APPLE dealer to get your copy of the additional card info – it'll be available about the time that you read this.



BITS AND PIECES

APPLE software bank contributors . . . please take note

Document, document, document. It would be difficult for you to give us too much information about a program that you send us – we need to know as much (and sometimes more) about your program contribution as you can tell us. And never assume that potential users of your program are as clever as you; instead, assume that they know nothing – not even when to hit RETURN! Remember: The care you take in explaining your program is the only guarantee that it will be usable by anyone else.

SAVE, then SAVE again. SAVE your program twice, one recording after the other, on the same cassette. By doing this you give yourself, and us, a safety factor in that if our computer cannot read the first SAVE, perhaps it will be able to read the second. (As you know, not all recorders are created equal.)

By the way, please add your name and address to the program listing with REM statements before you save it, so that there will be no question as to whose program we're running when we try it out.

LOAD, check, then give yourself credit. After you SAVE your program, turn OFF the Apple to clear the memory space you've been using, then power it up and LOAD the program to be sure that it does in fact load and run.

Finally, protect. After saving your program, make sure that it cannot be erased. Every cassette includes a write/protect feature in the form of two small plastic tabs on the edge of the cassette opposite its business end. When these tabs are pushed in, pulled out, or otherwise gotten out of the way, the cassette can no longer be written into.

Clearing the air. We make no profit from your contributions to the user portion of the APPLE software bank. We set up this portion of the bank simply to encourage and to ease the exchange of programs among APPLE users. The honorarium that we pay for contributions to this portion of the bank is merely our way of encouraging such contributions.

From time to time, of course, APPLE does purchase software, and does so at the fair market value. Such purchases are negotiated individually, based upon market conditions, applicability of the product, etc.

OUT OF THE MIST....

. . . ASCII character equivalents

The following table presents decimal-number equivalents and APPLE keyboard equivalents to ASCII (American Standard Code for Information Interchange) characters. That is, if you were to scan the keyboard directly in BASIC, these are the characters that you would read; or, if you were to go into memory, these characters would appear as string-variable values.

In the APPLE keyboard column, read S_MC as a SHIFT/CONTROL-M B^C as CONTROL-B, etc. The decimal numbers listed are those before clearing of the keyboard strobe; after clearing of the strobe, the number is the listed value minus 128 (e.g., after clearing of the strobe, 145 becomes 17). If an APPLE keyboard equivalent doesn't exist, or if there is no decimal equivalent, then the table shows a double dash (--); if the keyboard equivalent is identical to the ASCII character itself, then the table shows "sa.".

Such information can make your programming life much easier, because it lets you get keyboard data directly into a BASIC program without the use of an INPUT statement. As you know, INPUT statements can be limiting as, for example, when you type in a line and the screen yells SYNTAX ERROR at you. Well, what *is* the error? It may take a long time to find.

If, on the other hand, the keystrokes are picked directly off the keyboard then you – the programmer – are in command every step of the way; you make the decisions as to what goes and what does not, and in a way that tells you exactly *what's* wrong as you go wrong.

You can see one way in which this idea is applied in "Being Precise in INTEGER," in this issue's HOW TO section. In this Multiple Precision Arithmetic listing, statements 2500–2610 input data directly from the keyboard; they tell you exactly what's happening, the nature of any error you may have committed, etc. – and all by making use of the equivalence in the table presented here.

TABLE OF ASCII CHARACTER VALUES

ASCII CHAR	APPLE KYBD	DEC EQUIV	ASCII CHAR	APPLE KYBD	DEC EQUIV	ASCII CHAR	APPLE KYBD	DEC EQUIV	ASCII CHAR	APPLE KYBD	DEC EQUIV
NUL	S_PC	128	SP	Space Bar	160	@	sa.	192		--	224
SOH	A^C	129	!	sa.	161	A		193	a	sa.	225
STX	B^C	130	"		162	B		194	b		226
ETX	C^C	131	#		163	C		195	c		227
EOT	D^C	132	$		164	D		196	d		228
ENQ	E^C	133	%		165	E		197	e		229
ACK	F^C	134	&		166	F		198	f		230
BEL	G^C	135	'		167	G		199	g		231
BS	←	136	(		168	H		200	h		232
HT	I^C	137	)		169	I		201	i		233
LF	J^C	138	*		170	J		202	j		234
VT	K^C	139	+		171	K		203	k		235
FF	L^C	140	,		172	L		204	l		236
CR	M^C	141	–		173	M		205	m		237
SO	N^C	142	.		174	N		206	n		238
SI	O^C	143	/		175	O		207	o		239
DLE	P^C	144	0		176	P		208	p		240
DC1	Q^C	145	1		177	Q		209	q		241
DC2	R^C	146	2		178	R		210	r		242
DC3	S^C	147	3		179	S		211	s		243
DC4	T^C	148	4		180	T		212	t		244
NAK	U^C	149	5		181	U		213	u		245
SYN	V^C	150	6		182	V		214	v		246
ETB	W^C	151	7		183	W		215	w		247
CAN	X^C	152	8		184	X		216	x		248
EM	X	153	9		185	Y		217	y		249
SUB	Y^C	154	:		186	Z		218	z		250
ESC	sa.	155	;		187	[	--	219			251
FS	--	156	<		188	\	--	220			252
GS	S_MC	157	=		189	]	S_M	221	ALT	--	253
RS	S_NC	158	>		190	↑	--	222		--	254
US	--	159	?		191		--	223	DEL RUBOUT	--	255

LF = Line Feed; CR = Carriage Return; SP = SPace; ESC = ESCape; sa. = Keyboard character same as ASCII character.

HOW TO ...A moving experience

(Adapted from the Apple Core Newsletter, San Francisco, CA).

Ordinarily, APPLE displays only Page 1 of its memory (locations 1024 to 2047). But it is possible to display Page 2 (locations 2048 to 3071) as well; and if you know how to do it, use of Page 2 will give you black screens in a hurry, and snap your graphics and/or text material cleanly on and off.

Before you can use Page 2, however, you must tell APPLE not to put any variables at locations lower than 3072; in other words set LOMEM:3072. After you've done this, you're free to move the contents of Page 1 to Page 2, reload Page 1 with new data, and switch back and forth between the two pages. Here's how to do it, using the general-purpose block movement routines built into Apple's monitor.

POKE 60, (old starting address mod 256)
POKE 61, (old starting address / 256)
POKE 62, (old ending address mod 256)
POKE 63, (old ending address / 256)
POKE 66, (new starting address mod 256)
POKE 67, (new starting address / 256)
CALL -468 (the actual move command)

Now, to use Page 2 (remember to set LOMEM to 3072 or higher):

```
10  POKE 60,0:POKE 61,4:POKE 62,255:POKE 63,7:POKE 66,0:POKE 67,8: CALL -468: POKE -16299,0
```

To switch back and forth between Pages 1 and 2:

POKE -16299,0 (displays Page 2)
POKE -16300,0 (displays Page 1)

If both pages contain similar graphics figures, then switching between the pages will yield simple animation; further effects may be gleaned from an inspection of the list of POKEs on page 30 of the APPLE II Reference Manual. *(NOTE: Don't try this with APPLESOFT in RAM. It starts at hex 800 — the second page of graphic space. A block move into that area will send your APPLESOFT BASIC into the bit bucket!)*

The name of the game is the saving of the name

If you're an APPLESOFT II user working with, say, an inventory list with names, then you're in a bit of trouble if you want to SAVE the complete list to cassette tape, names and all: APPLESOFT II will save the numbers but not the names (strings). (Of course, the nicest way to SAVE such a list is to disk.)

If you need to save strings to tape however, the following program will do the job very nicely. Note that statement 10 creates space for the strings; 1010 gives you information about free memory space and how far up the variables are; 1050 writes out the length of the tape's string area; and 1070 writes all the desired information to tape.

```
PR#0


]

?SYNTAX ERROR
]LIST

1  REM
2  REM PROGRAM TO SAVE STRINGS
3  REM  TO CASSETTE TAPE.
4  REM BY R.WIGGINTON (6/78)
5  REM
10  DIM A$(10)
20  PRINT "TYPE IN NINE STRINGS, SEPARATED BY": PRINT "CARRIAGE RETURNS."
30  FOR K = 1 TO 9: INPUT A$(K): NEXT K
40  REM NOW SAVE A$ TO TAPE.
50  GOSUB 1000
55  PRINT "STRINGS ARE NOW ON TAPE.  TO RECALL,    TYPE 'GOTO 100', REWIND
    AND START TAPE, AND PRESS 'RTN'."
57  PRINT "LET TAPE RUN UNTIL CURSOR RETURNS."
60  END
100  REM THIS PART RECALLS THE
101  REM STRINGS FROM TAPE.
102  REM
110  DIM B$(10)
120  GOSUB 2000
130  FOR K = 1 TO 9: PRINT B$(K): NEXT K
140  END
1000  REM STORE A$ TO TAPE.
1003  PRINT "INSERT CLEAN TAPE, START RECORDING."
1005  PRINT "HIT ANY KEY WHEN READY": GET Z$
1010 X =  FRE (0): STORE A$: REM STORE A$ REALLY STORES POINTERS
1020  REM IN ORDER FOR THIS PROGRAM TO WORK, HIMEM MUST BE AT THE SAME
1021  REM VALUE WHEN THE STRINGS ARE RECALLED AS WHEN THEY ARE STORED.
1030 X =  PEEK (115) +  PEEK (116) * 256 -  PEEK (111) -  PEEK (112) * 256

1040  GOSUB 2100
1050  POKE 30,X -  INT (X / 256) * 256: POKE 31,X / 256: CALL  - 307: REM
    PUT (X) INTO LOCS 30&31, AND WROTE IT TO TAPE.
1060  REM (X) IS THE LENGTH OF THE STRING AREA.
1070  POKE 60, PEEK (111): POKE 61, PEEK (112): POKE 62, PEEK (115): POKE
    63, PEEK (116): CALL  - 307
1080  REM HAVE NOW WRITTEN EVERYTHING.
1090  PRINT "O.K.": RETURN
2000  RECALL B$: REM   GOT POINTERS BACK.
2010  GOSUB 2100: CALL  - 259: REM GOT LENGTH OF STRING AREA
2020 X =  PEEK (30) +  PEEK (31) * 256: REM XISLENGTH OF AREA TO READ IN
2030 X =  PEEK (115) +  PEEK (116) * 256 - X
2040  POKE 60,X -  INT (X / 256) * 256: POKE 61,X / 256
2050  POKE 62, PEEK (115): POKE 63, PEEK (116): CALL  - 259
2060  RETURN
2100  POKE 60,30: POKE 61,0: POKE 62,31: POKE 63,0: RETURN : REM  SET CASS
    ETTE ROUTINE POINTERS.

]
```

Lining things up, point by point

Since most BASICs justify (i.e., line up) to the left-most column, a display of multi-digit, decimal-pointed numbers can be awkward to read and somewhat unattractive.

A solution to this problem would be to use a tabulation routine that "justifies (or lines up) on the decimal point." Such a routine would position the numbers in a column so that the decimal points are vertically aligned. The short program listed below does exactly that.

Statements 10 through 50 are merely a demonstration routine that yields the sample run shown at the end of the listing. Statements 2000 through 2140 contain the routine that actually does the work. In effect, the routine aligns the numbers by right-justifying to the digit left of the decimal point, then tacks on the decimal point and the remaining digits to the right of the decimal point.

```
LIST
   10 F=-10:B=9:D=9
   15 A=F
   20 GOSUB 2000: PRINT ".";D
   30 IF A>3000 THEN 1000
   40 F=10*F:D=D+A
   50 GOTO 15
 1000 END
 2000 REM  RIGHT-JUSTIFICATION ROUTINE
      FOR APPLE BASIC
 2010 REM   INPUT IS ASSUMED TO BE IN
      VARIABLE "A"
 2020 REM  THE RIGHTMOST CHARACTER WILL
      APPEAR IN COLUMN CONTAINED IN
      VARIABLE "B"
 2100 A$=" ": IF A<0 THEN A$="-": REM
      GET SIGN OF NUMBER
 2110 A= ABS (A): REM  CONVERT A TO POS. #
 2120 C=(A>10)+(A>100)+(A>1000)+(A>10000):
      REM  DETERMINE HOW FAR TO
      LEFT-SHIFT PRINTOUT
 2130 TAB (B-C-1): PRINT A$;A;:
      REM  RIGHT-JUSTIFY PRINTOUT
 2140 RETURN
>RUN
       -10.9
      -100.19
     -1000.119
    -10000.1119
```

Being precise in INTEGER

The use of INTEGER BASIC limits you to the range of numbers between -32767 and +32767. Such a limitation is, at its best, frustrating, and, at its worst, infuriating. Consider, for example, the businessman who daily deals with foreign currencies, for which the basic monetary unit may be very, very small. What's a fella' to do?

Well, what he has to do is to go to multiple-precision arithmetic by means of a routine such as we present here. While this example is for addition only, it is readily adaptable to subtraction, multiplication, and division by changes in statements 3000 through 3080. The program does its job on large numbers in the same way as we do it by long-hand arithmetic; that is, it operates on one digit at a time, then carries to the next, and so on.

This particular listing is long and slow, because we wanted to make it clear and easy to read so that you could see what's happening. You may modify it to run much faster.

Incidentally, you can get a better understanding of the program's operation by relating certain of its statements to the ASCII conversion table that is in this issue's OUT OF THE MIST section. Statements 2500 through 2520, for example, result in the keyboard being read directly. 2540 refers to CHAR = 141; reference to the table tells you that decimal 141 is actually the Carriage Return. Similarly, 2545 excludes all characters except for the digits 0 through 9 (176 through 185). Again, statement 2550 converts the ASCII characters to the numbers themselves (i.e., if CHAR = 181, then 181 - 176 = 5).

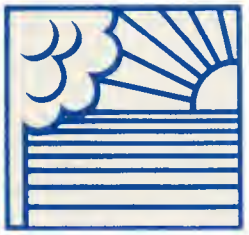
```
<** SYNTAX ERR
>LIST
   10 REM  MULTIPLE PRECISION ARITHMETIC
   20 REM  AN INTEGER BASIC EXAMPLE
   30 REM  THAT PROVIDES 20-DIGIT
   40 REM  ARITHMETIC PRECISION
   50 REM ------------------------------
  100 GOSUB 1000: REM  INITIALIZE EVERYTHING
  200 GOSUB GETA: REM  GET FIRST NUMBER INTO MATRIX A
  300 GOSUB GETB: REM  GET SECOND NUMBER INTO MATRIX B
  400 GOSUB ADDITION: REM  ADD MATRICES -- C=A+B
  500 GOSUB PUTC: REM  PRINT RESULT
  600 END
 1000 REM  INITIALIZATION ROUTINES
 1005 REM  ------------------------
 1010 DIM A(30),B(30),C(30),D(30),E(30)
 1020 GETA=2007:GETB=2100:PUTC=4010
 1030 ADDITION=3010
 1090 RETURN
 2000 REM  GET A ROUTINE
 2005 REM  ---------------------
 2007 PRINT "INPUT A";: TAB 20
 2010 GOSUB 2500: REM  GET INPUT INTO MATRIX E
 2020 FOR I=1 TO 30:A(I)=E(I): NEXT I: REM  MOVE NUMBER INTO A
 2030 RETURN
 2100 PRINT : REM  GET B ROUTINE
 2103 REM  ------------------------
 2105 PRINT "INPUT B";: TAB 20
 2110 GOSUB 2500: REM  GET INPUT INTO MATRIX E
 2120 FOR I=1 TO 30:B(I)=E(I): NEXT I: REM  MOVE NUMBER INTO B
 2130 RETURN
 2500 REM  KEYBOARD INPUT ROUTINE
 2505 REM  ------------------------
 2510 FOR I=1 TO 30:E(I)=0: NEXT I:CHAR=0:PTR=1: GOTO 2530
 2520 CHAR= PEEK (-16384): REM  READ KEYSTROKE
 2530 POKE -16368,0: REM  CLR LAST KEYSTROKE
 2540 IF CHAR=141 THEN 2590: REM  GOT C/R
 2545 IF (CHAR<176 OR CHAR>185) THEN 2520: REM  NOT A NUMBER, SO IGNORE IT
 2550 D(PTR)=CHAR-176: REM  CONVERT ASCII TO NUMBER AND SAVE
 2555 PRINT D(PTR);:PTR=PTR+1
 2560 IF PTR<21 THEN 2570: PRINT "INPUT TOO LONG--START OVER": POP : POP
      END
 2570 CHAR=0: GOTO 2530: REM  WAIT FOR NEXT KEYSTROKE
 2590 PTR=31:I=1: REM  WRAP UP  TRANSFER 0 INTO E
 2595 IF PTR-I<0 THEN RETURN
 2597 E(PTR-I)=D(PTR-I):I=I+1: GOTO 2595: REM  E=D, RIGHT-JUSTIFIED
 2610 RETURN  REM  LEAVE INPUT ROUTINES
 3000 REM  ADDITION ROUTINES
 3005 REM  ---------------------
 3010 CARYIN=0
 3020 FOR I=30 TO 1 STEP -1
 3030 CARYOUT=0:TEMP=A(I)+B(I)+CARYIN: REM  ADD COLUMN PLUS CARYIN
 3040 IF TEMP<10 THEN 3060
 3050 CARYOUT=CARYOUT+1:TEMP=TEMP-10: GOTO 3040
 3060 C(I)=TEMP:CARYIN=CARYOUT: REM  FINISHED ADJUSTING CARRY FIGURES
 3070 NEXT I
 3080 RETURN
 4000 REM  OUTPUT ROUTINE
 4005 REM  --------------------
 4010 PRINT : PRINT : PRINT
 4020 I=1
 4030 IF A(I)<>0 THEN 4040:I=I+1: GOTO 4030: REM  IGNORE LEADING ZEROS
 4040 TAB I+8: FOR J=I TO 30: PRINT A(J);: NEXT J
 4050 I=1
 4060 IF B(I)<>0 THEN 4070:I=I+1: GOTO 4060: REM  IGNORE LEADING ZEROS
 4070 PRINT : TAB I+8: FOR J=I TO 30: PRINT B(J);: NEXT J
 4080 I=1
 4090 IF C(I)<>0 THEN 4100:I=I+1: GOTO 4090: REM  IGNORE LEADING ZEROS
 4100 PRINT : TAB I+7: FOR J=I TO 31: PRINT "-";: NEXT J: PRINT
 4110 TAB I+8: FOR J=I TO 30: PRINT C(J);: NEXT J
 4115 PRINT : PRINT
 4120 PRINT : PRINT "****************************************": PRINT
      RETURN

>
 RUN
 INPUT A             9876543210
 INPUT B             9999999999


                          9876543210
                          9999999999
                         ------------
                         19876543209

 ****************************************
```

apple computer inc.
10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010



OUTSIDE THE ORCHARD

...Personal software

The G-2 line of personal-computer software is a series of program packages, with each package consisting of a cassette and full documentation. The first programs offered are "Beat the House", a Las Vegas-like game set that includes Blackjack, Craps, Slot Machines, and Roulette; "Dollars and Sense", a personal finance program; and "Clinic", a medical program that offers information on longevity, biorhythms, and dieting. Standard and Extended BASIC packages are also available. GRT Corporation, 1286 Lawrence Station Rd., Sunnyvale, CA 94086. (408) 734-2910.

Clock interface

Designed specifically for APPLE II, this real-time clock interface features on/off control via software; a time display anywhere on the screen as desired; time setting and testing from BASIC or machine language; relocatable software; and a slot-independent plug-in PC board. $32.95. System Design Engineering, Suite 40, 2460 W. 239th St., Torrance, CA 90505 (213) 539-2194.

6502 software

An organization for the exchange of tested software for 6502-based systems has been set up. The company does not buy copyrights, but pays royalties for and sells the software. Because APPLE is a 6502-based system, the programs may be of interest to APPLE users. For information, write to The 6502 Program Exchange, 2920 Moana, Reno, NV 89509.

Business programs

Here is a group of programs oriented to the small businessman and independent professional. The programs offer mailing list and secretarial services, time and data management, tax planning, daily scheduling, and even an exercise program. Each program runs in 16K or less of memory, and includes documentation and a security function that allows identification of unauthorized copies of the software tape. Retail prices range from $19.95 (the exercise program) to $89.95 (the tax planning and professional secretary packages). Charles Mann & Associates, 1926 South Veteran Ave., Los Angeles, CA 90025. (213) 473-0244.